

**REMARKS/ARGUMENTS**

Claims 1-50 are pending in this application, all of which have been rejected. Claims 27-37 have been rejected under 35 U.S.C. § 101. Claims 1-4, 13-17, and 21-30 have been rejected under section 102(e) as being anticipated by U.S. Patent No. 6,237,135 to Timbol. Claims 5-10, 34-39, and 43-50 have been rejected under 35 U.S.C. § 103(a) as being unpatentable over Timbol in view of U.S. Patent No. 6,424,979 to Livingston. Claims 11, 12, 18-20, 31-33, and 40-42 have been rejected under 35 U.S.C. § 103(a) as being unpatentable over Timbol in view of Livingston and in further view of U.S. Patent No. 6,370,531 to Boutcher. Various objections to the drawings, specification, and title have been raised.

In response to the Office Action, amendments have been made to the title, to various portions of the drawings and specification, and to claims 27 and 50.

**The Objections to the Drawings, Specification, and Title**

Paragraphs 3 and 4 of the Office Action state various objections to the drawings. In response to these objections, applicants have amended the drawings and specification, and applicants submit that these amendments are fully responsive to the objections. In particular: FIG. 12 has been amended to include reference numbers 1201-1204; the specification has been amended to refer to element 195 (which appears in FIG. 1), and element 1001 (which appears in FIG. 10).

Paragraph 3 of the Office Action requests that FIG. 4 be amended to change "410(1)" and "410(n)" to "401(1)" and "401(n)" respectively. Instead, applicants have amended the paragraph

of the specification appearing at page 15, lines 12-18 to use the reference numerals 410(1) and 410(n) shown in the drawings. It should be noted that numerals 410(1) and 410(n) are used in other portions of the specification to refer to the sequential code depicted in FIG. 4. Applicants submit that this change to the specification is fully responsive to the object to FIG. 4, in lieu of an amendment to FIG. 4.

Paragraph 4 of the Office Action indicates that elements 303, 304, and 305 are not referred to in the specification. This objection is incorrect. The Examiner is referred to page 17, line 26 through page 18, line 5 of the specification, wherein these reference numerals are referred to and described.

The title has been amended in accordance with the Examiner's suggestion at paragraph 5. Applicants note that the Examiner's suggested title has been adopted in the interest of expediting prosecution, and the new title does not reflect any change to the scope of the invention.

Paragraph 6 of the Office Action states an object to the Abstract, but appears to refer to lines of the specification. Applicants have thus treated paragraph 6 as an objection to the specification, and respectfully submit that the amendment to the specification addresses all of the issues raised in paragraph 6.

#### The Section 101 Rejection

Paragraph 7 rejects claims 27-37 under 35 U.S.C. § 101. In accordance with the Examiner's suggestion, applicant has amended claim 27 to recite that the database is on "one or

more computer-readable media.” No new matter has been added. Applicants submit that claim 27 is now allowable under section 101. Moreover, since claims 28-37 are dependent, either directly or indirectly, on claim 27, applicants submit that the amendment to claim 27 also addresses the section 101 rejection of claims 28-37.

### The Section 102 and 103 Rejections

Claims 1-50 have been rejected as being either anticipated by, or obvious over, the prior art Timbol, Livingston, and Boutcher references. Applicants respectfully submit that the Examiner has overlooked various claim features that are not taught or suggested by these prior art references. Below, applicants first briefly describe the present application and prior art cited, and then highlight certain claim features that distinguish the present invention over the prior art. In particular, applicants respectfully submit that the following is a non-exhaustive list of claim features that are neither taught nor suggested by the cited references, or by any combination of the references cited:

- A public object model with identifiable references to one or more events;
- Looking up code in a database based on a query;
- Software indexed or retrieved based on environmental features;
- A database that provides a pointer to a code module;
- A database that stores code remotely from the software object that executes the

code; and

- Selecting one of a plurality of code modules, with each code module having executable components corresponding to events.

The present application and the cited prior art

The present application is generally directed to the customization of software. In one embodiment described in the specification, a customized application includes an application object and a customization. The application object performs a set of fixed actions, and also fires “events,” which signify that certain points in the execution of the application object have been reached. The events may be received by the customization, which can then respond to the events by executing custom code. The custom code may be stored in a database, so that the database can be queried to find the right custom code for a given set of circumstances. For example, different users (or users from different departments) may have different sets of custom code. Different pieces of custom code may be stored in a database and indexed by a “moniker string,” which labels a given code module so that it can be located. (See Application, p. 3 line 2 through p. 4 line 24.)

The Examiner has cited the Timbol, Livingston, and Boutcher references in the rejection of the claims under sections 102 and 103. These references are briefly described below.

The Timbol patent is entitled “Development System with Visual Design Tools for Creating and Maintaining Java Beans Components.” The Timbol patent is specifically directed to rapid application development (RAD) using Java Beans, and the detailed description contained in the Timbol patent is focused on a “wizard,” which comprises a particular set of interfaces.

(See col. 7, line 41 through col. 19, line 60 & FIGS. 2C through 11.) A developer uses the wizard's interfaces to specify information about a particular "bean," and the wizard automatically creates the bean (See Timbol, Abstract.)

The Livingston patent is entitled "System for Presenting and Managing Enterprise Architectures." The system described in Livingston provides content to a user through a web browser. (See Livingston, col. 2, ll. 34-62.) A user is associated with a "profile," which indicates "the level of detail and time frame of information to be obtained." (See Livingston, Abstract.) Information satisfying the criteria specified in the profile is obtained from a database and then formed into a web page. (See Livingston, Abstract.)

The Boutcher patent is entitled "Method and Computer Program Product for Automatic Conversion of Data Based on Location Extension." The disclosure in Boutcher is narrowly focused on a Universal (or "Uniform") Naming Convention (UNC), which is used to identify data stored on a computer system. (See Boutcher, Abstract.) The bulk of the detailed description in Boutcher is dedicated to describing this naming convention (see Boutcher, FIG. 3 and col. 2, line 66 through col. 3, line 67).

As demonstrated below, none of the prior art cited teaches or suggests the features of the claims. Certain claim features that define over the prior art are individually discussed below.

A public object model with identifiable references to one or more events

Claim 1 calls for two object: (1) a base object, and (2) a customization object. The base object includes "a public object model which includes identifiable references to said one or

more events,” and the customization object includes “data or logic representative of said public object model.” As discussed above, the invention provides for events to be signified so that custom code can be invoked in response to the events, and the events are made publicly known so that third parties can write code that is invoked in response to the events. The public object model of claim 1 embodies this feature.

The Examiner has rejected claim 1 as being anticipated by Timbol, and proposes to read all of the features of claim 1 onto the following short text passage from Timbol:

The present invention provides a wizard-based tool which automatically generates code to define property setters and getters, accessor methods, event listener and registration mechanisms, and the like.

[Timbol, col. 9, ll. 64-67 (see Office Action, p. 4).] It is unclear how this portion teaches the existence of two object, or the use of a public object model that interrelates the two objects in the manner claimed. The Examiner has provided no explanation as to how this teaching anticipates claim 1. Applicants submit that, at a minimum, the cited portion does not teach the use of both a base object and a customization object, does not teach that the base object includes a public object model that includes identifiable references to one or more events, and does not teach that the customization object includes data or logic representative of the public object model. As described above, Timbol’s wizard comprises visual interfaces that allow a developer to specify features of a Java Bean, so that the wizard can automatically generate the bean. This “wizard” has nothing to do with the use of a “base object”, a “customization object,” the signifying of events, and the use of a public object model through which the base object and customization object are interrelated.

Thus, applicant submits that claim 1 is not anticipated by Timbol, and that the rather terse reading of claim 1 that the Examiner has proposed overlooks nearly every feature of claim 1. Thus, the rejection of claim 1 should be withdrawn.

Looking up code in a database based on a query

Independent claim 17 calls for “generating a database query” and “retrieving, based on said database query, a code module from a database.” Independent claim 27 similarly calls for a “plurality of custom code modules” that can be stored in a database and retrieved based on a query. Independent claim 38 calls for first and second sets of computer-executable instructions to be stored in a database, where either the first or second set is invoked based on which one satisfies a query. Dependent claims 5 and 10 recite similar features. These features are not taught or suggested by any of the prior art cited by the Examiner.

Independent claims 17 and 27 have been rejected under section 102 as being anticipated Timbol. The grounds for rejection of claims 17 and 27 are stated in paragraph 9 of the Office Action, and it should be noted that paragraph 9 does not specifically address the feature of storing code in a database, or retrieving the code based on a query. Instead, the Examiner merely cites and quotes various portions of Timbol, without any explanation as to how the cited portions teach the feature of storing code in a database. The rejection of claim 27 appears to apply Timbol to the language of claim 1; however, claim 1 does not recite the feature of code retrieved from a database based on a query, so it is unclear how the rejection of claim 1 applies to claim 27. The cited portions of Timbol do not teach or suggest the feature of

retrieving code from a database based on a query, and the Examiner has not provided any explanation of how those features can be derived from Timbol. In fact, at a subsequent point in the Office action the Examiner appears to admit that Timbol does not disclose retrieving code from a database based on a query (“Timbol failed to disclose supporting information concerning the database and queries thereon,” see ¶ 11), so it is unclear how the Examiner can maintain that Timbol anticipates claims 17 and 27. Thus, the section 102 rejection of claims 17 and 27 is incorrect and must be withdrawn.

With regard to claims 5 and 10 the Examiner does not assert that Timbol teaches the feature of retrieving code from a database. (As noted above, the Examiner admits this feature is not present in Timbol.) Instead, rejects these claims under section 103 as being obvious over Timbol in view of Livingston, and finds in Livingston the feature of looking up code in a database based on a query. (See Office Action, pp. 6-7.) The Examiner’s argument is that Livingston teaches the storage of information in a database, and it would be obvious to add Livingston’s database to Timbol “because components are typically stored, accessed through databases ....” The cited passages of Livingston, however, teach the storage of “content,” not code, in a database. As discussed above, Livingston is directed to the generation of web page content based on a user profile, and is not directed to the storage of custom code modules in a database. Applicants do not dispute that the use of a database to store data is in the prior art. However, claims 5 and 10 do merely call for the storage of data in a database, but specifically call for code to be stored in a database, and then for that code to be retrieved from the database



and invoked. No such system or method is described or suggested in the prior art. The mere fact that Livingston teaches that information can be stored in a database does not make it obvious to store code in a database, retrieve that code based on a query, and then execute the code that satisfies the query. Neither Timbol nor Livingston contains any suggestion as to why such use of a database would be desirable. Thus, the features recited in claims 5 and 10 are not obvious over Timbol in view of Livingston, and the section 103 rejection of claims 5 and 10 must be withdrawn.

Additionally, with respect to claim 38, the Examiner has rejected this claim under section 103 as being obvious over Timbol in view of Livingston. However, the Examiner does not appear to apply any particular portion of Livingston to claim 38, and instead relies entirely on Timbol (see Office Action, p. 8). As discussed above, neither Timbol nor Livingston teaches or suggests the use of a database to store code, or the retrieval of code from a database based on a query. Nor do Timbol and Livingston teach or suggest the invocation of a first or second set of computer-executable instructions depending upon which set satisfies a query, as called for in claim 38. Thus, the section 103 rejection of claim 38 should be withdrawn.

Software indexed or retrieved based on environmental features

Independent claim 44 calls for “ascertaining one or more attributes of said operating environment external to said software object” and “generating a database query based at least in part on said one or more attributes.” Claim 44 also recites that a code module, or a pointer thereto, is retrieved based on the query. Thus, claim 44 effectively calls for a particular code

module (or pointer thereto) is retrieved based on environmental attributes that are external to a software object that queries the database. Dependent claims 7, 12, 20, 32, and 50 (as amended) also recite that information that is used to query a database for a code module is derived from aspects of the environment in which the querying software operates. The present application describes these “environmental” aspects by way of example: see FIG. 8, showing that some data that forms the basis for a query may be based on information specific to the current user – e.g., the user’s identity and department – thereby allowing a code module to be retrieved based on which user is logged in, based on which department that user is in, or based on some other environmental circumstance. In this vane, claim 33 recites that a “moniker string” is used to identify a custom code module, and that the moniker string is based on “the identity of a user or organization.” None of these features are taught in the prior art cited by the Examiner.

Claims 7, 44 and 50 have been rejected under section 103(a) as being obvious over Timbol in view of Livingston. In support of the rejection of claims 44 and 50, the Examiner merely cited the portion of Timbol that states: “The present invention provides a wizard-based tool which automatically generates code to define property setters and getters, accessor methods, event listener and registration mechanisms and the like” (see Office Action, p. 8). Nothing in this passage describes the use of a query that is based on information derived from the environment in which software operates. In support of the rejection of claim 7, the Examiner cites a portion of Livingston that describes the storage of information in a database, and another portion of Livingston that mentions “rules” and a “style sheet.” Again, neither of these portions teaches or suggests that a query is derived from aspects of the environment (or the use of this query to

retrieve a code module from a database). Moreover, the Examiner has not explained how this claim feature can be derived from Timbol, Livingston, or any combination thereof. Thus, the rejection of claims 7, 44, and 50 is based on an incorrect reading of Livingston and/or Timbol, and should be withdrawn.

Claims 12, 20, 32, and 33 have been rejected under section 103(a) as being obvious over Timbol in view of Livingston and in further view of Boutcher. In support of the rejection of claims 12, 20, and 32, the Examiner has cited a portion of Boutcher that addresses a file-naming convention. While the appearance of Boutcher's file names bears a superficial resemblance to the moniker strings of the present application (e.g., both have fields separated by the slash ("/" character), claims 12, 20, and 32 are not merely directed to the use of moniker strings, but also to the use of information that is derived from aspects of the environment, as described above. Apparently the Examiner is referring to the fact that, in one example, the information derived from the environment becomes part of the moniker string that is used to label a code module, and is later used to lookup that code module (see application, FIG. 8). The fact that Boutcher's file names look somewhat like the moniker strings described in the present application does not explain how Boutcher teaches or suggests deriving information from aspects of the environment, and the use of this information to retrieve code from a database. Thus, it appears that the Examiner's attempt to apply claims 12, 20, and 32 to Boutcher overlooks various features of those claims. Applicants thus submit that the section 103(a) rejection of claims 12, 20, and 32 is incorrect and should be withdrawn.

In support of the rejection of claim 33, the Examiner has cited col. 18, ll. 10-55 of Livingston (see Office Action, p. 9). As noted above, claim 33 calls for a moniker string that is based on environmental information including the identity of a user or organization. The cited portion of Livingston describes a “naming convention.” While the naming convention described in Livingston is quite intricate, nowhere does it teach (or even suggest) a string that is based on environmental data including the identity of a user or organization, as called for by claim 33. Nor has the Examiner explained any motivation to modify the cited references to yield this feature. Thus, the rejection of claim 33 overlooks the feature of deriving a moniker string from environmental information including the identity of a user or organization, and thus the rejection of claim 33 must be withdrawn.

A database that provides a pointer to a code module

Claim 21 calls for a database that “provides a pointer to [a] code module,” and for the code module to be retrieved by “following said pointer.” Claim 21 is dependent on independent claim 17, and, like claim 17, has been rejected under section 102 as being anticipated by Timbol. As discussed above in connection with claim 17, Timbol teaches the storage of content in a database. Just as Timbol does not teach the storage of code in a database (see discussion of claim 17 above), Timbol also does not teach the storage of pointers to code in a database. There is no teaching in Timbol of a symbol that retrieves a pointer to a code module from a database, and then retrieves the code module by following the pointer.

The above described features have been overlooked by the Examiner in applying Timbol to claim 21. Thus, the finding that Timbol anticipates claim 21 is incorrect, and the section 102 rejection of claim 21 must be withdrawn.

A database that stores code remotely from the software object that executes the code

Claim 36 is dependent, through a series of claims, on claim 27. Claim 27 calls for a software object that requests a code module from a database, receives the code module, and then loads the code module for execution. Claim 36 calls for the database to be “located remotely from a computing device that executes said software object.” The Examiner has rejected claim 36 under section 103(a) as being obvious over Timbol in view of Livingston. In support of this rejection, the Examiner asserts (see Office Action, p. 7) that Timbol’s teaching of a “non-visual bean that runs on a server” corresponds to the claimed remote database.

Apparently, the Examiner assumes (without explicit support) that the server on which Timbol’s “non-visual bean” runs is located “remotely” to some software object. However, this apparent reading misconstrues claim 36. Claim 36 does not call for a “bean” (or any type of software) that runs remotely on a server. Rather, claim 36 calls for a “software object” that receives a code module from a database, *where the database is located remotely from the software object*. In other words, Timbol (according to the Examiner) teaches that code is executed remotely, while claim 36 calls for code to be retrieved from a remote location but executed locally. The cited teaching of Timbol merely calls for a “non-visual bean” to be run on a “server,” which neither teaches nor suggests the claimed feature.

Moreover, there is no teaching or suggestion in any of the cited art that code can be loaded for execution after having been received from a remotely located database. Thus, the rejection of claim 36 clearly overlooks this feature, and this rejection must be withdrawn.

Selecting one of a plurality of code modules, with each code module having executable components corresponding to events

Independent claim 9 is directed to a method of performing a task that includes fixed actions and variable actions. According to claim 9, one custom code module is selected from a plurality of custom code modules. Each custom code module comprises one or more executable components, where each component corresponds to a particular variable action. For each variable action, an event is signified. Then, a particular component is invoked from the selected code module, depending upon which variable action corresponds to the signified event. Claim 9 has been rejected as being obvious over Timbol in view of Livingston. However, neither Timbol nor Livingston teaches or suggests any of these features.

In support of the rejection of claim 9, the Examiner generally cites Timbol as disclosing “a component based, application development system to included [sic] customizable objects” (see Office Action, p. 6), and Livingston as disclosing “XML data,” “attributes,” “an expiration date,” and “an e-mail reminder” (see Office Action, p. 7). It is unclear how the Examiner finds in these teachings any of the above-described features of claim 9. Timbol and Livingston do not teach custom code modules that contain executable components; nor do they teach the feature of each executable component corresponding to a particular variable action; nor do they teach the

feature of a particular component being invoked based on which variable action corresponds to the signified event. The Office Action contains no explanation as to how Timbol and Livingston teach or suggest these features, and applicants submit that these features are not taught or suggested by any of the prior art cited.

Thus, the section 103(a) rejection of independent claim 9 appears to overlook several features that are not taught or suggested in any of the cited prior art, and the rejection of claim 9 should be withdrawn.

#### The Dependent Claims

As demonstrated above, independent claims 1, 9, 17, 27, 38, 44, and 50 (as well as the various dependent claims discussed above) explicitly recite features that are neither taught nor suggested by the prior art. Since all of the claims that are not specifically discussed above are dependent, either directly or indirectly, on the claims that have been shown to define over the prior art, the claims that have not been specifically discussed are patentable at least by reason of their dependency. Thus, claims 1-50 are patentable over the prior art, and the section 102 and 103 rejections of those claims should be withdrawn.

#### Amendment to claim 50

Claim 50 has been amended to recite that “the code module being selected [is] based, at least in part, on aspects of an environment in which said code module executes, said aspects being external to an object that causes said code module to be invoked.” No new matter has

**DOCKET NO.:** MSFT-0231/160306.1  
**Application No.:** 09/678,511  
**Office Action Dated:** April 25, 2003

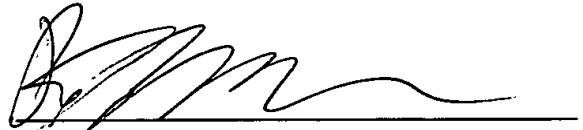
**PATENT**

been added by this claim. The amendment finds support in at least FIGS. 10 and 11, and the text descriptive of those figures (page 22, line 18 through page 23, line 23).

### **CONCLUSION**

For all of the foregoing reasons, applicants respectfully submit that this case is now in condition for allowance, and an early notice of allowance is earnestly solicited.

Respectfully submitted,

A handwritten signature in black ink, appearing to be 'P. M. Ullman', written over a horizontal line.

Peter M. Ullman  
Registration No. 43,963

Date: September 19, 2003

Woodcock Washburn LLP  
One Liberty Place - 46th Floor  
Philadelphia PA 19103  
Telephone: (215) 568-3100  
Facsimile: (215) 568-3439